NASA SEWP Security Center

Using Sender ID in a Linux Environment

Erika McCallister
Scott He
September 16, 2005

DISCLAIMER

This document is intended for informational purposes only. It represents the NASA SEWP Security Center's experiences using Sender ID in a Linux environment. There are no express or implied warranties regarding the veracity of the information provided. There are no guarantees that your experiences with Sid-milter, Fedora, or Sendmail will be similar.

Table of Contents

1 Overview of Sender ID

Sender ID is a tool that was created to prevent spam and email forgery. It uses the DNS to authenticate the origin of email. Each time a mail server receives an email, it extracts the domain of the sender and the originating IP address from the message. The mail server then checks the information against a special record in the DNS. The special record is called an SPF record, and it contains a list of IP addresses permitted to send email from the particular domain. If the information extracted from the email message matches the SPF record, then the message is delivered to the appropriate mailbox. Information that does not match the SPF record can be handled in several ways based on the policies of the receiver.

Sender ID is the result of a compromise between two existing authentication protocols. Although both protocols work as described above by checking the DNS, the protocols differ based on how they determine the correct sender domain. Microsoft created a protocol called Caller ID. Caller ID relied on the Purported Responsible Address (PRA), which is usually defined as the “RFC 2822 sender” field. This sender field is the “from” field within the message header. In some circumstances, the PRA is defined to be another field (see section 1.2). In contrast, PO Box created the Sender Policy Framework (SPF) protocol, which uses the “RFC 2821 sender” field. The “RFC 2821 sender” field is the “MAIL-from” field in the SMTP protocol data. The advantage to SPF is that the sender can be authenticated before the message is sent and parsed. However, SPF has a higher rate of false positives than Caller ID. SPF is still used as a stand-alone authentication method.

Microsoft provided an outline of the basic steps of how Sender ID works:

1. The sender transmits an e-mail message to the receiver.
2. The receiver's inbound mail server receives the mail.
3. The inbound server checks which domain claims to have sent the message, and checks the DNS for the SPF record of that domain. The inbound server determines if the sending e-mail server's IP address matches any of the IP addresses that are published in the SPF record.
4. If the IP addresses match, the mail is authenticated and delivered to the receiver. If the addresses do not match, the mail fails authentication and is not delivered.

1.2 Algorithm for PRA

The PRA is usually the “from” field of the message header. However, is some cases, it is defined to be other fields, such as the “resent-sender” header.

The PRA algorithm has the following steps:

1. Select the first non-empty Resent-Sender header in the message. If no such header is found, continue with step 2. If it is preceded by a non-empty Resent-From header and one or more Received or Return-Path headers occur after said Resent-From header and before the Resent-Sender header, continue with step 2. Otherwise, proceed to step 5.

2. Select the first non-empty Resent-From header in the message. If a Resent-From header is found, proceed to step 5. Otherwise, continue with step 3.

3. Select all the non-empty Sender headers in the message. If there are no such headers, continue with step 4. If there is exactly one such header, proceed to step 5. If there is more than one such header, proceed to step 6.

4. Select all the non-empty From headers in the message. If there is exactly one such header, continue with step 5. Otherwise, proceed to step 6.

5. A previous step has selected a single header from the message. If that header is malformed (e.g. it appears to contain multiple mailboxes, or the single mailbox is hopelessly malformed, or the single mailbox does not contain a domain name), continue with step 6. Otherwise, return that single mailbox as the Purported Responsible Address.

6. The message is ill-formed, and it is impossible to determine a Purported Responsible Address.

2 Installing Sendmail and Sid-milter on Fedora Core 4

As Linux newbies, our task of creating a network test environment for the new Sid-milter email filter for Sendmail was quite daunting. We installed our Fedora Core 4 cds and ventured forth into the unknown. It wasn't completely unknown; we were familiar with the basics (i.e. rmdir, ls, cd, etc.). However, our greenness with Linux was pretty apparent right away as we struggled with the topic of disabling unnecessary services and preventing them from restarting upon reboot. We conquered this task, mastered NMAP, and we were ready to install Sendmail.

You may ask why we would install a fresh version of Sendmail when Fedora comes pre-loaded with Sendmail. The answer is Libmilter. The default install of Sendmail does not include this library, including the magical libmilter.a file, which is required to run any milter (aka mail filter). You'll realize quickly that the magical file is missing if you receive about 300 error messages when trying to install a milter.

Once you have your first box ready to go, you may want to clone it to create more boxes for a test network. We discovered that the use of LVM in Fedora Core 4 made using

Norton Ghost 2003 nearly impossible. We finally gave up and just manually installed the OS and software for the other three machines in our network.

We downloaded the newest stable version of Sendmail (8.13.4). The first step we recommend is reading the instructions. I repeat, read the instructions. This simple approach eluded us for a while. The instruction file is aptly entitled, "README." This file will greatly reduce your level of confusion. Here are the steps we took to install Sendmail while ensuring that libmilter built and installed correctly. There is one caveat. Each time we installed Sendmail, we received slightly different errors, even though we used the same procedure on identical machines. Be ready for anything. We'll try to cover all of the issues we encountered. Some issues merely reflect our amateur status in the use of configuration files. On the bright side, we learned a lot from this process and no longer feel like complete newbies.

2.1 Sendmail Installation for use with Sid-milter

1) Download Sendmail 8.13.4 from http://www.sendmail.org
2) Make sure you have a good chunk of time available.
3) Make sure that your box is recognized on the network by adding it to the DNS.
4) Uninstall Sendmail if it is already installed. We prefer the yum method:
    a. # yum remove Sendmail
5) Download Sendmail 8.13.4 at sendmail.org
    a. Extract the package to usr/local
6) From the new Sendmail directory, build libmilter:
    a. # cd /usr/local/sendmail/libmilter
    b. # sh Build
    c. # sh Build install
7) Do a quick search for libmilter.a. This will save you headaches later.
    a. # find / -name libmilter.a –print
8) Build the mini-parts of Sendmail. For example:
    a. # cd /usr/local/sendmail/vacation
    b. # sh Build
    c. If you get the "cannot find usr/man/man1/vacation.1" error, just create the directories and copy the vacation.1 file into it. You may see more errors of this type, but we assure you that simply building the directories and copying the file will do the trick.
        i. If you want to get fancier, you can go through the sendmail.mc file and specify the correct directory. We learned this after the fact.
    d. Repeat this build process for other sub-directories, such as praliases. This helps to prevent the vacation-type errors later.
9) Here's the tricky part. The instructions call for you to find the appropriate pre-made configuration file and build it. We did this using the "generic-linux.mc" file, but it didn't have enough stuff to make Sendmail operate correctly with Sid-milter. Instead, we prefer the yum method. It won't mess up your libmilter, but it will provide you with the best config file. This will sound illogical, but it works:

a. Go ahead and follow the README instructions for the generic configuration.
    i. # cd /usr/local/sendmail/cf/cf
    ii. # cp generic-linux.mc sendmail.mc
    iii. # sh Build sendmail.cf
    iv. # sh Build install-cf
b. Go ahead and do the big build on Sendmail as a whole.
    i. # cd /usr/local/sendmail
    ii. # sh Build install
    iii. As the build information is passing down the screen, keep an eye out for libmilter to ensure it's there.
c. Now add the yum magic:
    i. # yum install Sendmail
d. Yum doesn't cause any problems to your current build, but it adds nice config files.
    i. Delete the old config files in /etc/mail
    ii. Go back to sendmail/cf/cf
    iii. You should have some new files called sendmail.mc.rpmsomething and sendmail.cf.rpmsomething
    iv. Rename the files so that .rpmsomething is gone
        1. # mv sendmail.mc.rpmsomthing sendmail.mc
        2. # mv sendmail.cf.rpmsomething sendmail.cf
e. You need to change the sendmail.mc file so that it will work with the milter:
    i. Add these two lines to the end of the file:
        1. # gedit sendmail.mc
        2. type:
        3. INPUT_MAIL_FILTER(`sid-filter', `S=inet:xxxx@localhost')
        4. define(`confINPUT_MAIL_FILTERS', `sid-filter')
    ii. Special notes:
        1. xxxx refers to the port number of your choice. This will be where sid-filter connects with Sendmail. Be sure to choose a port not associated with another service. We chose 8891.
        2. Notice the parameters begin with ` (back tick) and end with ' (single quote).
    iii. If you plan to have your mail sent to and from other machines, then you need to make three more modifications to this file.
        1. Find dnlTRUST_AUTH_MECH(`digest-md-5 cram-md-5 Login plain')dnl
            a. Remove the dnl from the beginning so that the line is not ignored.
        2. Find dnldefine('confAUTH_MECHANISMS', `DIGEST-MD-5…PLAIN)dnl
            a. Remove the dnl from the beginning so that the line is not ignored.
        3. Find DAEMON_OPTIONS(`Port=smtp, Addr=127.0.0.1, Name=MTA')dnl

        a. Add dnl to the beginning of this line so that Sendmail listens for external connections.
    4. Rebuild the modified configuration file:
        a. # cd /usr/local/sendmail/cf/cf
        b. # sh Build sendmail.cf
        c. # sh Build install-cf
10) Now, you are ready for the final build.
    a. # cd usr/local/Sendmail
    b. # sh Build –c
    c. # sh Build install
11) Sendmail should be ready to use
    a. # service sendmail start
12) Send a test message to yourself
    a. create a text file named test
    b. send the file to yourself
        i. # mail root < test
    c. Check for the message
        i. # mail
        ii. You should see the message
        iii. Type the message number to read it

Another great reference for installing and configuring Sendmail is the Linux Home Networking site. Please see the references section.

2.2 The Berkeley Database Installation

If you plan to use the Sendmail features that require use of a database, such as creating virtual user accounts with virtusertable.db, then you may want to install the Berkeley Database. This section is optional, and there may be other packages available to accomplish these tasks.

1) Download the Berkeley Database from: http://www.sleepycat.com/products/db.shtml
2) Extract the package (we extracted to the desktop).
3) Load the instructions from the docs folder into your browser. Index.html is the start page.
4) Click on the “Building for Unix/Posix” systems link.
5) Follow the instructions listed:
    a. Change to the build_unix directory
        i. # cd /root/Desktop/BerkeleyDB-4.3.28.NC/build_unix
        ii. # ../dist/configure
        iii. # make
        iv. # make install
6) Now that the database is installed, you need to change the Sendmail configuration to work with the database.

a. Change to your Sendmail directory
    i. # cd /usr/local/sendmail/devtools/Site
    ii. Create a new site.config.m4 file
        1. # gedit site.config.m4
        2. Add the following lines:

APPENDDEF(`confINCDIRS', `-I/usr/local/BerkeleyDB.4.3/include')
APPENDDEF(`confLIBDIRS', `-L/usr/local/BerkeleyDB.4.3/lib')

        3. Save the new file
    iii. You need to rebuild Sendmail to add database support.
        1. Change to the source directory
            a. # cd ../..
            b. sh Build –c
            c. sh Build install
        2. If you receive error messages, check the name and location of the database files as specified in the lines you added to site.config.m4.
b. You now have database support for special mail server features.

2.3 Sid-milter Installation

1) Download Sid-milter 0.2.9 from: http://sourceforge.net/project/showfiles.php?group_id=112121
2) Extract to /usr/local
3) This release has a coding error, so you will need to modify the ar.c source file before building it.
    a. # cd /usr/local/sid-milter/libar
    b. # gedit ar.c
        i. Go to line 1041
        ii. Change ar_res_init() to res_init()
        iii. Save ar.c
4) The sid-milter is ready to build:
    a. First, build the subdirectory of sid-filter. This should help to prevent the "cannot find usr/man/man8/makemap.8" error. If you still get this error, or similar errors make the necessary directories or edit the configuration files as discussed above in section 8.c.
        i. # cd /usr/local/sid-milter/sid-filter
        ii. # sh Build
    b. Now, you're ready to build the entire milter
        i. # cd /usr/local/sid-milter
        ii. # sh Build
        iii. # sh Build install

2.4 Using Sid-filter with Sendmail

1) Before starting sid-filter, stop your Sendmail service.
    a. # service sendmail stop
2) Using sid-filter is relatively easy. There are a few options you can specify when running it.
    a. –l creates a logfile
    b. –p sets the port. Use the port you specified in the Sendmail configuration files.
    c. –t allows you to run the filter in test mode so that rejected emails are still received.
    d. –r lets you set an operating level that specifies how strict you want the filter to be:
        i. 0 is the default. It accepts all email.
        ii. 1 - reject the email if both tests fail (sender-id and spf).
        iii. 2 - reject the email if one test fails.
        iv. 3 - reject the email unless one test passes.
        v. 4 - reject the email unless both tests pass.
    e. An example:
        i. # sid-filter –l –r 4 –p inet:8891@localhost
        ii. In this example, we enabled logging, set the level to the most stringent, and set the port as 8891.
    f. For more information about these options and other options, please read the man page.
    g. Start Sendmail
        i. # service sendmail start
    h. You can check that the filter is working by sending yourself an email. The sid-filter header should appear in the header section of the email.

2.5 Using Sendmail with an MX Record

As a short note, we added an MX record for our domain and tested it with Sid-filter. We added the MX record to sewpsc.sewp.nasa.gov. The record listed the mail server as callisto.sewpsc.sewp.nasa.gov. We then used the virtual user table to create an alias for the root account of the sewpsc.sewp.nasa.gov domain. To do this, we added to our DNS record that callisto was the mail server and gave it the highest level of priority.

To create a working account for root@sewpsc.sewp.nasa.gov, we added the address to the virtusertable file in /etc/mail on the callisto box. We then had to convert the file to a database which can be read by Sendmail. This is an instance in which the Berkeley Database would be helpful.

1) # cd /etc/mail
2) # gedit virtusertable
3) Add the email address and the local account to which the mail should be send and save the file.
    a. root@sewpsc.sewp.nasa.gov root

4) Build the database from the file
    a. # makemap hash virtusertable < virtusertable

3 SPF Records

Sender-ID requires the use of SPF records in DNS. An SPF record is a DNS record of the type "text" (TXT). The general type TXT allows any information to be added about a particular domain. A TXT record that is being used as an SPF record requires a particular format so that it can be parsed correctly.

An SPF record includes the following:

Type	Description	Examples
Version	Version of SPF being used.	v=spf1, v=spf2.0
Scope	Tells Sender ID if the record matches PRA, mail-from address or both. If version 2 is designated, then scope must be present.	v=spf2.0/mfrom,pra
Mechanisms	Mechanisms describe the set of hosts designated to send email. They are evaluated from left to right. The evaluation results in one of three options: match, not match, or exception.	a:example.com ip4:127.0.0.1 mx
Modifiers	Key-value pairs providing additional information that affect evaluation. Always contains an "=" sign.	redirect=_example.com
Prefixes	Work with mechanisms to designate whether an IP address should pass or fail. If not designated, then + is implied.	+all ~all

Types of Mechanisms:

Mechanism	Description
all	Matches all local and remote IPs and goes to the end of the SPF record. Example:v=spf1 +all
include	Specifies other domains that are authorized domains. Example:v=spf1 include:domain.com -all
a	Specifies all IPs in the DNS A record. Example: v=spf1 a:domain.com -all
mx	Specifies all A records for each host's MX record. Example: v=spf1 mx mx:domain.com -all
ptr	Specifies all A records for each host's PTR record. Example: v=spf1 ptr:domain.com -all
ip4	Specifies a single IP or an acceptable IP address range. /32 is

	assumed if no prefix-length is included. (ip6 also works “ip6:”). Example:v=spf1 ip4:192.168.0.1/16 –all
exists	Specifies one or more domains normally singled out as exceptions to the SPF definitions. An A query is performed on the provided domain, if a result is found a match occurs. Example:v=spf1 exists:domain.com -all

The Meaning of Prefixes:

Prefixes	Description
+	Pass. The address passed the test. This is the default if not specified. Example:v=spf1 +all
-	Fail. The address failed the test. Example:v=spf1 -all
~	Softfail. The address failed the test, but the result is not definitive. Example:v=spf1 ~all
?	Neutral. The address did not pass or fail the test. Example:v=spf1 ?all

3.1 Examples

- v=spf1 mx ip4:192.168.100.136 –all
  - This example shows that we are using SPF version 1, we are testing the MX record for the domain in order of MX priority, and only the listed IP address is permitted to send email from the domain.
- v=spf1 a:example.com –all
  - This example again shows the use of SPF version 1, and it tests the A records for the domain are tested. The “-all” means that no other domains are permitted.
- v=spf1 ip4:192.168.0.1/16 –all
  - This example also uses SPF version1. It uses CIDR notation and permits any IP address between 192.168.0.1 and 192.168.255.255.
- v=spf1 include:example.net –all
  - This example again shows SPF version 1. The include mechanism allows you to cover other domains. This is good for situation where you use example.net, example.cc, and example.com. In this case, the other domain is checked for a match.
- v=spf2.0/pra mx ip4:122.654.100.2 ~all
  - This example uses SPF version 2, which must provide the scope. In this case, the scope is to only check the PRA. It checks the MX record and allows this one IP address. The tilde indicates that it is not known whether other permitted addresses exist, and it results in a soft-fail if an IP address does not match.
- v=spf1 +all

    - This example uses SPF version 1. It accepts all IP addresses. This example is not recommended for use.
- v=spf1 –all
    - This example also uses SPF version 1. This domain is not permitted to send mail.

4 Our Evaluation of Sender ID Using Sendmail with Sid-filter

We performed a lot of testing on Sender ID after actually getting everything installed and configured correctly. These tests ranged from regular usage to spoofing to checking the effect on our DNS server.

We created a fairly simple test network of five computers. Four of the boxes were mail servers, and one box was the domain controller, which ran the DNS using Windows Server 2003. Each computer was a Dell Optiplex Pentium 3 at 933mhz with 256 MB RAM, except for the domain controller. The mail servers all ran on Fedora Core 4. We installed Sendmail 8.13.4 on each of them. We installed Sid-milter 0.2.9 to work with Sendmail as the Sender ID filter. The mail servers were named Callisto, Europa, Ganymede, and Io. The test network was blocked from sending and receiving email from outside the network.

4.1 Basic Testing Sender ID

We created many types of SPF records in DNS and tested all four running levels of Sid-filter to determine whether emails would be accepted. Our test network consisted of four identical machines running Linux with Sendmail and Sid-filter. The box called Callisto was designated as the sender, and its domain had an SPF record that was modified for each test. The other three machines were used receivers and were set to run levels 1-4 on the Sid-filter. Sid-filter behaved as expected. The fail result from the filter caused Sendmail to reject the message at all levels, whereas a soft-fail only caused Sendmail to reject the message at the run levels 3 and 4.

Summary of Results:

SPF Record	Running Level Results				Authentication Header
	r=1	r=2	r=3	r=4	
v=spf1 mx a:io.sewpsc.sewp.nasa.gov ~a	accept	accept	reject	reject	soft-fail
v=spf1 mx a:io.sewpsc.sewp.nasa.gov -a	reject	reject	reject	reject	fail
v=spf1 mx ip4:177.28.31.66 ~all	accept	accept	reject	reject	soft-fail
v=spf1 mx ip4:177.28.31.66 -all	reject	reject	reject	reject	fail
v=spf1 mx a:callisto ~a	accept	accept	reject	reject	soft-fail
v=spf1 mx a:callisto -a	reject	reject	reject	reject	fail
v=spf1 mx ip4:192.168.100.136 ~all	accept	accept	accept	accept	pass

v=spf1 mx ip4:192.168.100.136 -all	accept	accept	accept	accept	pass
v=spf1a:callisto.sewpsc.sewp.nasa.gov ~all	accept	accept	accept	accept	pass
v=spf1a:callisto.sewpsc.sewp.nasa.gov -all	accept	accept	accept	accept	Pass
v=spf2.0/pra ip4:192.168.100.136 -all	accept	accept	reject	reject	neutral
V=spf2.0/mfrom ip4:192.168.100.136 –all	accept	accept	reject	reject	neutral
V=spf2.0/mfrom,pra ip4:192.168.100.136 –all	accept	accept	reject	reject	neutral
V=spf2.0/mfrom,pra ip4:192.168.100.136 ~all	accept	accept	reject	reject	neutral

Notes:

1) The sending machine was called Callisto, and it has an IP address of 192.168.100.136.
2) The IP address 177.28.31.66 was arbitrarily made up.
3) If no MX record is found, Sender ID defaults to the SPF record of the sender's domain. Thus, using "mx" as a mechanism will not affect the outcome if no MX record is present.
4) The sender was not Io.

4.2 The Effect of Sender ID on DNS

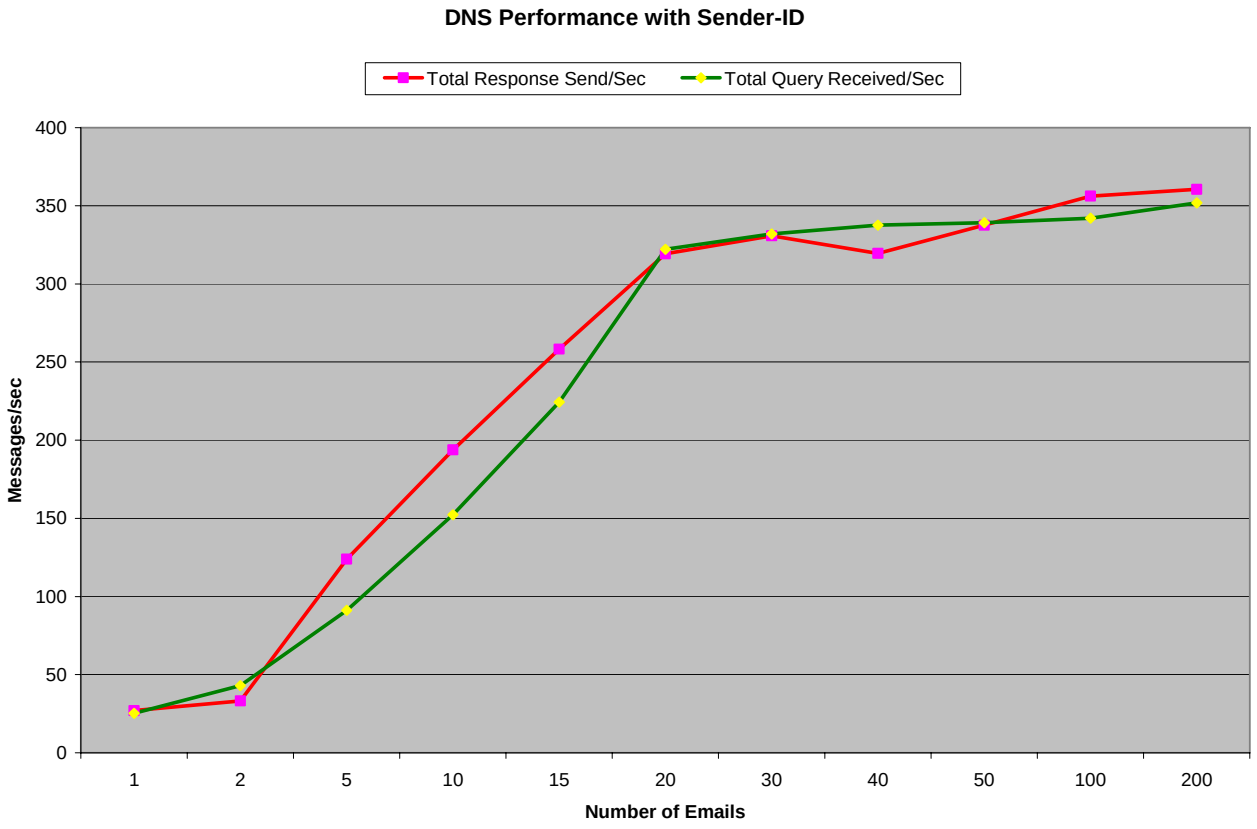
According to internet drafts describing Sender ID, it is highly dependent and reliant on the steady supply of requests and answers generated by the DNS server. In fact, any interruption or disruption of this relationship with DNS will stop Sender ID from functioning correctly. Specifically, the check_host() process in the Sid-filter sends UDP packets to the DNS to perform its tasks. Our interest was to monitor and study Sender ID's communication exchange to and from the DNS server. We used a Bash shell script to flood the DNS with Sender ID requests. We measured the effect of the requests using the performance monitoring tool that accompanies MS Server and delay indicator shown in the maillog.

First, we focused on the 'Total Response Sent Per Second' and 'Total Query Received Per Second' as monitored through our Windows DNS server. Second, we looked at the patterns of individual mail delays as generated in the mail-log (/var/log/maillog). Each run of the test raised the number of emails per pack and was repeated three times. The numbers were averaged to produce the final result. The email delays were sporadic but followed a trend of increasing time delay, as well as an increase in the number of delayed emails. For the batch of 200 emails, there were 19 delays with a maximum time of 24 seconds.

Both the 'Total Response Send/Sec' and 'Total Query Received/Sec' achieve a performance plateau beginning with 30 emails per batch at around 350 messages per second. Our theory is that this peak will be maintained as mail volume increases which will cause the DNS to process at this peak for a greater sustained period of time.

In conclusion, we do not believe that the DNS will be a significant bottleneck for Sender ID. DNS was designed to perform quick lookups and adding the Sender ID traffic did not seem to pose a threat to its performance capabilities.

Number of Emails	Total Response Send/Sec	Total Query Received/Sec	Max Delay
1	27	25.3	0
2	33.2	43	8
5	123.9	91.1	8
10	193.8	152.2	8
15	258.2	224.4	8
20	319.3	322.2	8
30	330.8	331.9	8
40	319.5	337.5	8
50	337.6	339.2	16
100	356.2	342	16
200	360.5	351.8	24



4.3 Forgery and Spoofing Tests

We made several attempts to spoof emails to test how well Sender ID does what it claims. Our first attempt involved a straightforward spoofing attempt using "# sendmail –bs" which allowed us to open an interactive SMTP session. We set the receiving machine to a Sid-filter run level of 2. We created an SPF record for the spoofed

machine that permitted only the IP address for that box. We sent the message from a third box using the interactive mode. The receiving machine properly rejected the message as a forgery.

Here is the interactive SMTP session:

```
[root@ganymede ~]# sendmail -bs
220 ganymede.sewpsc.sewp.nasa.gov ESMTP Sendmail 8.13.4/8.13.4/Submit; Wed, 31 Aug
2005 14:37:11 -0400
HELO localhost
250 ganymede.sewpsc.sewp.nasa.gov Hello root@localhost, pleased to meet you
MAIL from: <root@callisto.sewpsc.sewp.nasa.gov>
250 2.1.0 <root@callisto.sewpsc.sewp.nasa.gov>... Sender ok
RCPT to: <root@io.sewpsc.sewp.nasa.gov>
250 2.1.5 <root@io.sewpsc.sewp.nasa.gov>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
from: root@callisto.sewpsc.sewp.nasa.gov
to: root@io.sewpsc.sewp.nasa.gov
subject: hi
Tricked!
.
250 2.0.0 j7VIbB6B025821 Message accepted for delivery
QUIT
221 2.0.0 ganymede.sewpsc.sewp.nasa.gov closing connection
```

Here are the Sid-filter results from the Maillog:

```
Aug 31 14:39:03 io sendmail[9578]: j7VId3AC009578:
from=<root@callisto.sewpsc.sewp.nasa.gov>, size=620, class=0, nrcpts=1,
msgid=<200508311837.j7VIbB6B025821@ganymede.sewpsc.sewp.nasa.gov>,
proto=ESMTP, daemon=MTA, relay=ias1.sewpsc.sewp.nasa.gov [192.168.100.138]
Aug 31 14:39:03 io sendmail[9578]: j7VId3AC009578: Milter insert (1): header:
Authentication-Results: io.sewpsc.sewp.nasa.gov
from=root@callisto.sewpsc.sewp.nasa.gov; sender-id=fail (NotPermitted);
spf=fail (NotPermitted)
Aug 31 14:39:03 io sendmail[9578]: j7VId3AC009578: Milter: data, reject=554 5.7.1
Command rejected
Aug 31 14:39:03 io sendmail[9578]: j7VId3AC009578:
to=<root@io.sewpsc.sewp.nasa.gov>, delay=00:00:00, pri=30620, stat=Command
rejected
```

Next, we decided to test an obvious spoof of a message allegedly from Hotmail, since Hotmail has already implemented Sender ID, and it has an active SPF record. The message was sent using "# sendmail –bs." The message claimed to be from heather11@hotmail.com, which was chosen randomly. The receiving machine caught the message as a forgery and added "soft-fail" to the authentication header because the SPF record ended in ~all. If the receiving machine had been using a higher run level, then it would have rejected the message. Therefore, Sender ID seems to have accomplished its most basic objectives in the prevention of forged email.

4.3.1 Forging Return Addresses for Bounced Messages

Spammers have recently begun using the return address as a means to get spam sent through filters. Basically, spammers insert the intended recipient's address into the MAIL-FROM and REPLY-TO fields, and then send the message to a non-existent email account. The email then bounces back to the intended recipient. Ideally, Sender ID should reject these messages because the MAIL-FROM and PRA domains do not authorize the sending IP address.

We tested the Sender ID against the bounced spam technique, and it failed to reject the incoming bounced messages. Again, we used "# sendmail –bs" for an interactive SMTP session. The sending machine Europa sent the message to a non-existent user "toto" on another machine called Ganymede. The "MAIL-from:", "from:", and "reply-to" fields were set as "root@sewpsc.sewp.nasa.gov." The record for sewpsc.sewp.nasa.gov had an MX record listing Callisto as the mail server. The SPF record for Callisto only authorized Callisto's IP address. Likewise, the other machines in the test also only permitted mail from their own IP address. The authentication header on the bounced emailed was "sender-id=pass."

Here is the interactive SMTP session:

```
[root@europa log]# sendmail -bs
220 europa.sewpsc.sewp.nasa.gov ESMTP Sendmail 8.13.4/8.13.4/Submit; Thu, 1 Sep
2005 15:04:11 -0400
HELO localhost
250 europa.sewpsc.sewp.nasa.gov Hello root@localhost, pleased to meet you
MAIL from: <root@sewpsc.sewp.nasa.gov>
250 2.1.0 <root@sewpsc.sewp.nasa.gov>... Sender ok
RCPT to: <toto@ganymede.sewpsc.sewp.nasa.gov>
250 2.1.5 <toto@ganymede.sewpsc.sewp.nasa.gov>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
from: root@sewpsc.sewp.nasa.gov
reply-to: root@sewpsc.sewp.nasa.gov
to: toto@ganymede.sewpsc.sewp.nasa.gov
hi hi
.
250 2.0.0 j81J4BwB020615 Message accepted for delivery
QUIT
221 2.0.0 europa.sewpsc.sewp.nasa.gov closing connection
```

The Sid-filter results for the bounced message based on the Maillog:

```
Sep  1 15:03:43 callisto sendmail[31180]: j81J3htW031180: from=<>, size=2553,
class=0, nrcpts=1,
msgid=<200509011906.j81J66Cx020634@europa.sewpsc.sewp.nasa.gov>, proto=ESMTP,
daemon=MTA, relay=europa.sewpsc.sewp.nasa.gov [192.168.100.143]
Sep  1 15:03:43 callisto sendmail[31180]: j81J3htW031180: Milter insert (1): header:
Authentication-Results: callisto.sewpsc.sewp.nasa.gov from=MAILER-
DAEMON@europa.sewpsc.sewp.nasa.gov; sender-id=pass
```

```
Sep  1 15:03:43 callisto sendmail[31182]: j81J3htW031180:
to=<root@sewpsc.sewp.nasa.gov>, delay=00:00:00, xdelay=00:00:00, mailer=local,
pri=32911, dsn=2.0.0, stat=Sent
```

Interestingly, if the message is instead sent to a fictitious address at Callisto, then only the SPF part of Sender ID returns a “fail” result.

```
[root@europa log]# sendmail -bs
220 europa.sewpsc.sewp.nasa.gov ESMTP Sendmail 8.13.4/8.13.4/Submit; Thu, 1 Sep
2005 15:41:13 -0400
HELO localhost
250 europa.sewpsc.sewp.nasa.gov Hello root@localhost, pleased to meet you
MAIL from: <root@callisto.sewpsc.sewp.nasa.gov>
250 2.1.0 <root@callisto.sewpsc.sewp.nasa.gov>... Sender ok
RCPT to: <toto@callisto.sewpsc.sewp.nasa.gov>
250 2.1.5 <toto@callisto.sewpsc.sewp.nasa.gov>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
from: root@callisto.sewpsc.sewp.nasa.gov
reply-to: root@callisto.sewpsc.sewp.nasa.gov
to: toto@callisto.sewpsc.sewp.nasa.gov
subject: hi
this is a test from Europa.
.
250 2.0.0 j81JfDHE020962 Message accepted for delivery
QUIT
221 2.0.0 europa.sewpsc.sewp.nasa.gov closing connection
```

Here are the authentication header results:

```
From MAILER-DAEMON@callisto.sewpsc.sewp.nasa.gov  Thu Sep  1 15:40:57 2005
Authentication-Results: callisto.sewpsc.sewp.nasa.gov from=MAILER-
DAEMON@europa.sewpsc.sewp.nasa.gov; sender-id=pass; spf=fail (NotPermitted)
```

In this instance, the message was rejected at a run level four.

4.4 An Interesting Quirk

An interesting problem we encountered was that emails sent and received on the same box resulted in a “fail” status and were rejected depending upon the run level. The problem is that internal mail never uses an external IP address, so when Sid-filter checks the SPF record, it determines that 127.0.0.1 does not match the record. We discovered that adding the IP address of localhost 127.0.0.1 to the SPF record worked as a temporary fix. The correct solution is to start Sid-filter with the –P option for peerlist. A peerlist is a list of IP addresses that you choose to exclude from being checked by the filter. You just need to create a list in a file and tell Sid-filter where it is located. Once we added localhost to this file, then the strange rejections ended. An example:

#sid-filter –l –r 4 –P /test/peerlist –p inet:8891@localhost

5 Limitations of Sender ID

Sender ID is not a silver bullet that will eliminate spam. It is simply another tool that will work with other anti-spam technologies to reduce the amount of spam and instances of forgery and phishing. Sender ID is not a complete solution to the problem of spam. For example, a spammer who uses a valid SPF record can still send spam that will not be stopped by Sender ID. On the other hand, it is extremely helpful in the prevention of phishing scams because a spammer cannot easily send forged emails purportedly from your bank asking for personal data, assuming your bank posts an SPF record in DNS. Sender ID is not a perfect anti-spam tool, and it has been criticized for several reasons.

A primary criticism of Sender ID is that it will generate many false positives and discard valid email because many messages are received from IP addresses that are not authorized by the sender. For example, mail that is automatically forwarded from one server to another is currently rejected by Sender ID. A concrete example of mail forwarding is when you change email addresses. You may want the old address to forward mail to your new address until everyone you know has been notified of the change. We tested this Sender ID problem by adding a .forward file in the /home directory of a user on one of our boxes. The receiving server rejected the email because the domain of the original sender A did not match the IP address of the forwarding sender B. A solution called Sender Re-writing Scheme (SRS) has been proposed, which would require the forwarding box to replace both sender fields with its email address. However, this would require an implementation change for all mail servers. Moreover, bounced messages would not be returned to the original sender. Finally, it could introduce more fraud into the system by allowing an email header to be changed in transit.

Another major criticism of Sender ID is that it is only as secure as DNS because it was built upon DNS. DNS was originally designed for simplicity and efficiency. Therefore, DNS has no inherent security features and can be exploited. For example, attackers can pose as the local DNS server and respond to requests with forged records, which would affect Sender ID. Additionally, a resourceful attacker could fake the IP address by hijacking the address space. An attacker can alter the IP router structure if the attacker has access to a router that is involved with external BGP routing. The attacker could then advertise a more specific route to a rogue SMTP client and override the legitimate owner of the address. Therefore, the results of a Sender ID lookup may be vulnerable to DNS exploits.

Another criticism is that Sender ID only authenticates the domain and not the actual sender. Thus, it is still possible to forge an internal address. For example, if bob@example.com pretends to be alice@example.com, then Sender ID will not catch

the forgery. Bob would still be sending the message from an authorized IP address for example.com.

Sender ID has also been criticized because it does not block self-replicating email viruses. When a virus sends email to every person in an address book, the messages will be accepted by the receiver because Sender ID just matches domain name to allowed server IP addresses. Thus, email sent by viruses would appear to be legitimate and would be accepted by Sender ID.

A final criticism of Sender ID of Sender ID is that you must trust all other domain owners to keep a valid and updated SPF record. To prevent frustration and hassle, many domain owners may just add "+all" to their SPF records, which would authorize all IP addresses. Authorizing all IP addresses is a tactic that could be used by spammers. Alternatively, legitimate domains that authorize all IP addresses could be targeted by spammers at potential victims of forgery. Domain owners should use SPF records to protect their goodwill and customers against spammers and phishers. Thus, Sender ID is dependent upon the quality of the SPF records.

6 References

[1] Advogato.org. *Why You Shouldn't Jump on the SPF Bandwagon*. January 13, 2005. http://www.advogato.org/article/816.html.

[2] Allman, E. & Katz, H. *SMTP Service Extension for Indicating the Responsible Submitter of an E-Mail Message*. IETF Internet Draft, May 2005. http://www.ietf.org/internet-drafts/draft-katz-submitter-01.txt

[3] Ipswitch.com. *User Guide: Setting Up and SPF Record*. Last Accessed August 29, 2005. http://www.ipswitch.com/support/ics/guides/IMailServer/8_2/IMailUGHTML/Chapter%202%20config12.html

[4] Linuxhomenetworkingguide.com. *Chapter 21: Configuring Linux Mail Servers*. Last Accessed July 22, 2005. http://www.linuxhomenetworking.com/linux-hn/sendmail.htm.

[5] Lyon, J. *Purported Responsible Address in E-Mail Messages*. IETF Internet Draft, May 2005. http://www.ietf.org/internet-drafts/draft-lyon-senderid-pra-01.txt

[6] Lyon, J. & Wong, M. *Sender ID: Authenticating E-Mail*. IETF Internet Draft, May 2005. http://www.ietf.org/internet-drafts/draft-lyon-senderid-core-01.txt.

[7] Messaging Anti-Abuse Working Group. *Important Considerations for Implementers of SPF and/or Sender ID*. July 11, 2005. http://www.maawg.org/about/whitepapers/spf_sendID/.

[8] Microsoft. *Sender ID Framework – Deployment Overview*. August 25, 2004. http://www.microsoft.com/downloads/details.aspx?familyid=8958AB23-F350-40FE-BA0A-2967B968FD8D%20&displaylang=en.

[9] Microsoft. *Sender ID Resources: Tools and Information about the Technology*. May 2, 2005. http://www.microsoft.com/mscorp/safety/technologies/senderid/resources.mspx.

[10] Microsoft. *Sender ID Framework: Implementation Tips for the Sender ID Framework – Creating your SPF record*. March 3, 2005. http://download.microsoft.com/download/1/1/8/1184dafa-f1c6-4cd6-8fa1-0b06abbebd79/spf_tips.pdf

[11] Sourceforge. Project: Sender ID Milter. Last Accessed September 13, 2005. http://sourceforge.net/projects/sid-milter/

[12] Wong, M. & Schlitt, W. Sender *Policy Framework (SPF) for Authorizing Use of Domains in Email*, version 1. IETF Internet Draft, June 6, 2005. http://www.ieft.org/internet-drafts/draft-schlitt-spf-classic-02.txt.